

FIA/P GRADUAÇÃO

ENTERPRISE APPLICATION DEVELOPMENT

Prof. Me. Thiago T. I. Yamamoto

#02 – C# E ORIENTAÇÃO A OBJETOS

TRAJETÓRIA



Plataforma .NET



Linguagem C# e Orientação a Objetos

#02 - AGENDA



- Linguagem C#
- Hello World
- Operadores básicos
- If, Switch e Loops
- Tipos de dados e modificadores de acesso
- Princípios da orientação a objetos
 - Abstração
 - Herança
 - Polimorfismo
 - Encapsulamento
- Enum, Interfaces e Listas
- Tratamento de exceções

- **Linguagem para criação de componentes .NET**

- Criação de aplicações, serviços e bibliotecas;
- Orientada a Objetos (POO);
- Sintaxe similar ao Java e C++

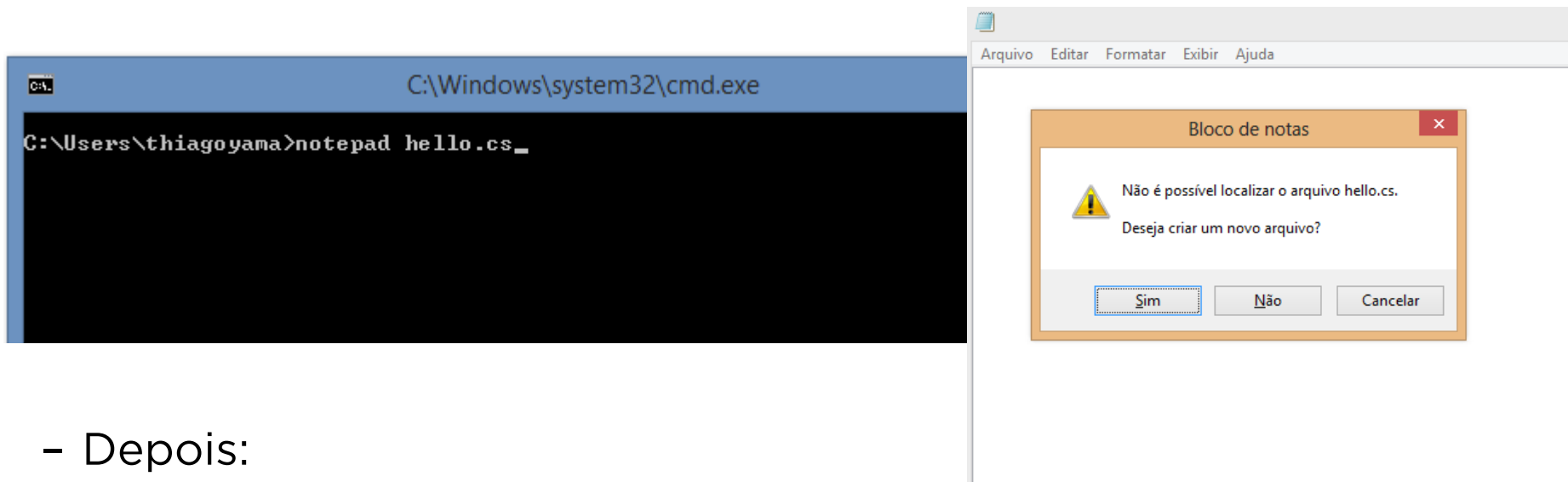
```
public static void Main(string[] args)
{
    if (DateTime.Now.DayOfWeek == DayOfWeek.Friday)
    {
        Console.WriteLine("Opa! Hoje é sexta!");
    }
}
```

- O compilador C#:
 - Transforma o código em C# na **Linguagem Intermediária**;
 - Produz um assembly (.dll, .exe)



HELLO WORLD!

- Vamos criar um **hello world sem** utilizar IDE (Visual Studio)
- Primeiro vamos criar um arquivo chamado **hello.cs** (.cs é a extensão para códigos escritos em C#)
 - Abra o **cmd do windows**, navegue até o diretório onde deseja criar o arquivo e digite: `notepad hello.cs`

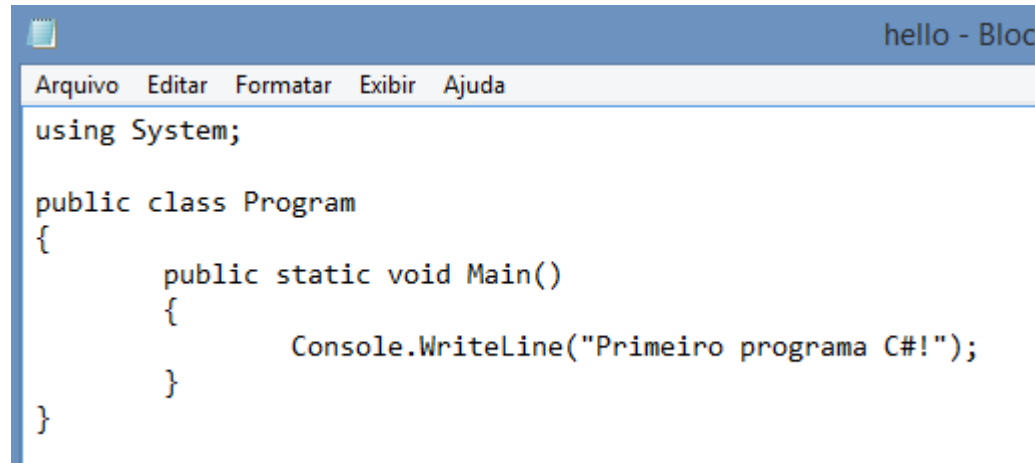


– Depois:

Sim!, você quer criar um novo arquivo!

HELLO WORLD!

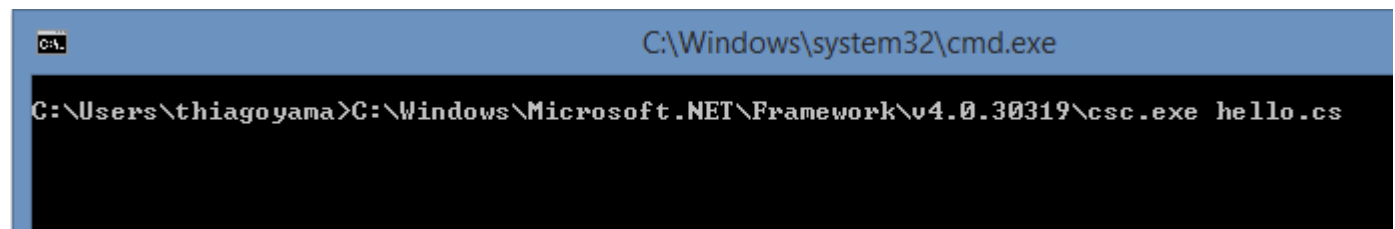
- No arquivo, digite o código abaixo:



```
using System;

public class Program
{
    public static void Main()
    {
        Console.WriteLine("Primeiro programa C#!");
    }
}
```

- Agora é hora de **compilar** o arquivo! Para isso, navegue até o diretório de instalação do .NET Framework, utilize o **csc.exe** para compilar arquivo passando o nome do arquivo como parâmetro:

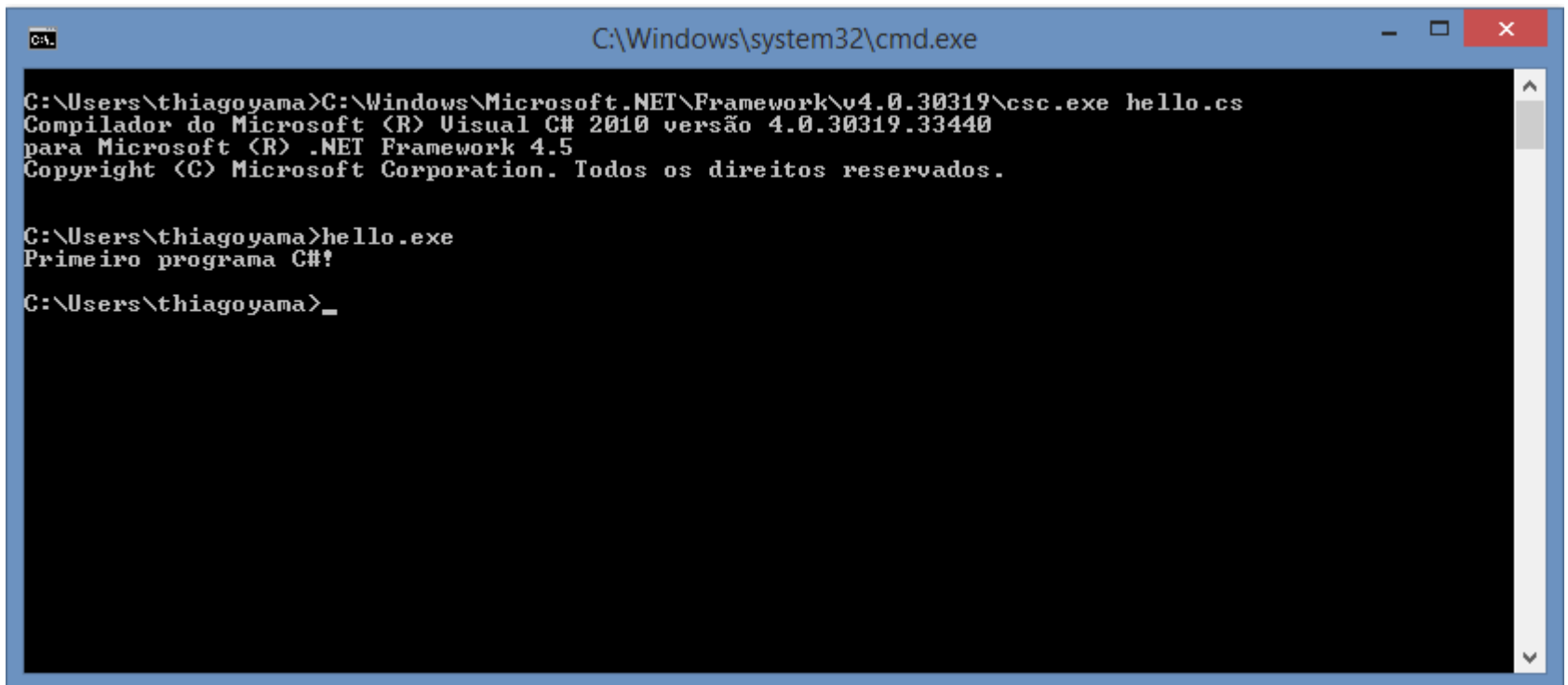


```
C:\Windows\system32\cmd.exe

C:\Users\thiagoyama>C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe hello.cs
```


HELLO WORLD!

- Foi gerado um arquivo **hello.exe**! (Executável)
- Agora é só executar e ver o resultado:



```
C:\Windows\system32\cmd.exe

C:\Users\thiagoyama>C:\Windows\Microsoft.NET\Framework\v4.0.30319\csc.exe hello.cs
Compilador do Microsoft (R) Visual C# 2010 versão 4.0.30319.33440
para Microsoft (R) .NET Framework 4.5
Copyright (C) Microsoft Corporation. Todos os direitos reservados.

C:\Users\thiagoyama>hello.exe
Primeiro programa C#!

C:\Users\thiagoyama>_
```



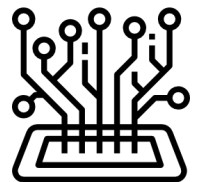
C#

- **Matemáticos:** +, -, *, /
- **Relacionais:** <, >, <=, >=, ==, !=
- **Lógicos:** &&, ||, !
- **Atribuição:** =, +=, -=, *=, /=



- Estrutura de seleção;

```
if (idade <= 18)
{
    //Código...
}
else
{
    //Código...
}
```



- Usado com integers, characters, strings e enums;
- Default é opcional;

```
switch (nome)
{
    case "Thiago":
        //Código..
        break;
    case "Maria":
        //Código
        break;
    default:
        //Código
        break;
}
```

```
for (int i = 0; i < 10; i++)  
{  
    //Código  
}
```

```
do  
{  
    j++;  
    //Código  
} while (j < 10);
```

```
while (j < 10)  
{  
    j++;  
    //Código  
}
```

Nome abreviado	Classe do .NET	Type (Tipo)	Width	Intervalo (bits)
byte	Byte	Inteiro sem sinal	8	0 a 255
sbyte	SByte	inteiro com sinal com sinal	8	-128 a 127
int	Int32	inteiro com sinal com sinal	32	-2,147,483,648 to 2,147,483,647
uint	UInt32	Inteiro sem sinal	32	0 a 4294967295
short	Int16	inteiro com sinal com sinal	16	-32.768 a 32.767
ushort	UInt16	Inteiro sem sinal	16	0 a 65535
long	Int64	inteiro com sinal com sinal	64	-922337203685477508 to 922337203685477507
ulong	UInt64	Inteiro sem sinal	64	0 a 18446744073709551615
float	Single	Tipo de ponto flutuante de precisão simples	32	-3.402823e38 para 3.402823e38
double	Double	Tipo de ponto flutuante de precisão dupla	64	-1.79769313486232e308 para 1.79769313486232e308
char	Char	Um único caractere Unicode	16	Unicode símbolos usados no texto
bool	Boolean	Tipo booliano lógico	8	True ou false
object	Object	tipo de base de todos os outros tipos		
string	String	Uma sequência de caracteres		
decimal	Decimal	Preciso tipo fracionário ou integral que pode representar números Decimal com 29 dígitos significativos	128	$\pm 1.0 \times 10e-28$ para $\pm 7.9 \times 10e28$

- Palavras chaves que declara o **nível de acesso**:

Palavra Chave	Aplicável para	Descrição
public	Class, Membros	Sem restrição
protected	Membros	Acesso para classe e seus filhos
internal	Class, Membros	Acesso para todos do mesmo Assembly
protected internal	Membros	Acesso para todos do mesmo Assembly e Classes filhas
private	Membros	Acesso somente para a classe

Membros: Métodos, Propriedades;



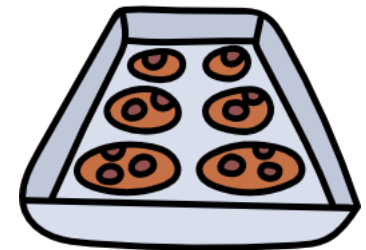
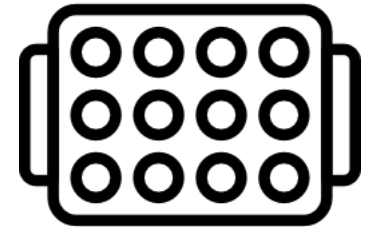
PRINCÍPIO DA ORIENTAÇÃO A OBJETOS

- **Classes define:**

- Estado;
- Comportamento;

- **Objeto é uma instancia de uma classe:**

- É possível criar várias instancias;
- Cada instância possui diferentes estados;



- Princípios da orientação a objetos:

Abstração

Herança

Encapsulamento

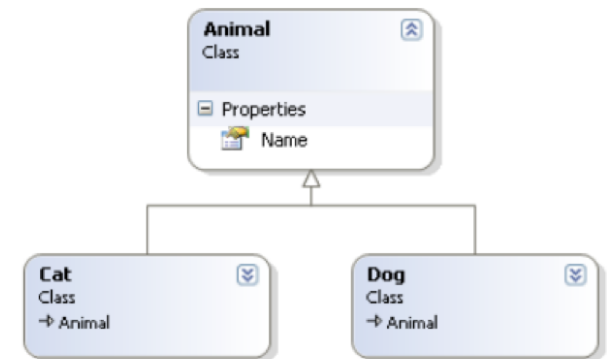
Polimorfismo

- Classes que **estendem** de outras classes:
 - Todas as classes descendem de **System.Object**;
 - Ganham todos os estados e comportamentos da classe base.

```
class Animal
{
    public string Name { get; set; }
}

class Dog : Animal
{
}

class Cat : Animal
{
}
```



- **Fields** são variáveis da classe:
 - Estáticos ou de instância;
- **Fields readonly:**
 - É possível atribuir valores somente na declaração ou construtor;

```
class Animal
{
    private readonly string _nome;

    private int _idade;

    public Animal(string nome)
    {
        _nome = nome;
    }
}
```

- Cada **propriedade** define get e/ou set;
- Usado para expor e controlar os **fields**;
- Nível de acesso para get e set são **independentes**;

```
private string _nome;  
  
public string Nome  
{  
    get { return _nome; }  
    set  
    {  
        if (!string.IsNullOrEmpty(value))  
        {  
            _nome = value;  
        }  
    }  
}
```

- Implementação **automática** de propriedades usando campo **oculto**;
 - Nível de acesso de get e set são independentes

```
public string Nome { get; set; }
```

Atalho: Digite **prop** e depois clique duas vezes tab, uma propriedade é criada automaticamente

- **Inicialização** de propriedades:

```
public class Cachorro
{
    public string Nome { get; set; }
    public string Raca { get; set; }
}
```

```
Cachorro dog = new Cachorro()
{
    Nome = "Duke",
    Raca = "Pug"
};
```

É possível **inicializar** as **propriedades** mesmo que a classe **não** tenha um **construtor** com argumentos;

Atalho: Digite **ctor** e depois clique duas vezes tab, uma propriedade é criada automaticamente

- **Base:** utilizado para chamar construtores ou métodos da classe pai;

```
public class Animal
{
    public string Nome { get; set; }

    public Animal(string nome)
    {
        this.Nome = nome;
    }
}
```

O **construtor** da classe filha (Cachorro) **deve chamar** um construtor do pai;

```
public class Cachorro : Animal
{
    public string Raca { get; set; }

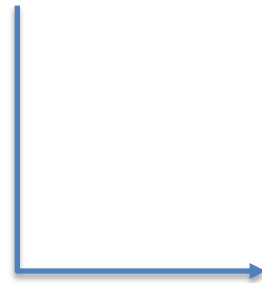
    public Cachorro(string nome, string raca): base(nome)
    {
        this.Raca = raca;
    }
}
```

- Palavra chave **abstract**: aplicado a classe e membros;
- **Classe Abstrata** não pode ser instanciado:
 - Define uma classe base para outras classes;
 - **Pode** implementar métodos abstratos (sem implementação);

```
public abstract class Animal
{
    public abstract void Eat();
}
```

Classe **Dog** herda de **Animal** e implementa o **método abstrato**;

```
public class Dog : Animal
{
    public override void Eat()
    {
        //Código
    }
}
```

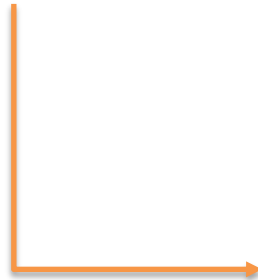


- Palavras chave **virtual**:
 - Permite **sobrescrever** o método da classe;

```
public class Animal
{
    public virtual void Eat()
    {
        //Código
    }
}
```

Classe **Dog**
sobrescreve o
método **Eat()** da
classe **Animal**

```
public class Dog : Animal
{
    public override void Eat()
    {
        //Código
        base.Eat();
    }
}
```





C# OUTROS ASSUNTOS

- Conjunto de **constantes**;
- Por padrão, o valor armazenado para cada constante é int;

```
public enum Genero
{
    Masculino,
    Feminino,
    Outros
}
```

- Define um grupo com mesmos métodos e propriedades:
 - Todos os membros são **públicos**;
 - **Classes** e structs podem herdar **várias interfaces**;
 - Por padrão, a **primeira letra** do nome da interface deve ser **I**

```
interface IMessage
{
    void SendMessage(string msg);
}
```

Classe
EmailMessage
implementa a
interface **IMessage**

```
class EmailMessage : IMessage
{
    public void SendMessage(string msg)
    {
        //Send email message code
    }
}
```



- Estrutura para trabalhar com **coleções** de variáveis:
 - Armazena o **mesmo tipo** de variável;
 - Começa com index **0**;

```
ICollection<String> lista = new List<String>();  
lista.Add("Thiago");  
lista.Add("Yamamoto");  
  
Console.WriteLine(lista[0]);
```

- Iterar uma coleção de itens:
 - Não podemos modificar a coleção durante a iteração;

```
foreach (var item in lista)
{
    Console.WriteLine(item);
}
```


- Herdam de **System.Exception**
- Podemos **lançar** e **tratar** exceções

```
if (age < 18)
{
    throw new ArgumentException("Menor de idade");
}
```

Algumas Exceções:

- ✓ System.DivideByZeroException
- ✓ System.IndexOutOfRangeException
- ✓ System.InvalidCastException
- ✓ System.NullReferenceException

- **Try, Catch e Finally;**
- **Finally:** executado sempre;

```
try
{
    CheckAge(17);
}
catch (ArgumentException)
{
    //Código
}
finally
{
    //Código
}
```

- É possível tratar **várias exceções** no mesmo bloco;
- A exceção mais **genérica** deve ser tratada no **final**;

```
try
{
    CheckAge(17);
}
catch (ArgumentException)
{
    //Código
}
catch (Exception)
{
    //Código
}
```

CODAR!

- Para praticar a **nova linguagem** de programação e aplicar os **princípios de orientação a objetos**, desenvolva o exercício **01 - C# e Orientação a Objetos**;



VOCÊ APRENDEU..



- Linguagem **C#**;
- **Operadores** básicos da linguagem;
- **Modificadores de acesso** e **tipos de dados**;
- Princípios da orientação a objetos: **abstração**, **polimorfismo**, **encapsulamento** e **herança**;
- Trabalhar com **enum**, **interfaces** e **listas**;
- Realizar o **tratamento** de **exceções**;

Copyright © 2013 – 2019
Prof. Me. Thiago T. I. Yamamoto

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).

“Aprender é a única coisa que a mente nunca se cansa, nunca tem medo e nunca se arrepende”